Enhancing Access Control in Distributed Systems Through Intelligent ABAC Policy Mining

Sudhir Kumar Bai, Jason Aaron Goveas and Barsha Mitra
Dept. of Computer Science & Information Systems, Birla Institute of Technology and Science, Pilani, Hyderabad Campus, India

Keywords: ABAC, Policy Creation, Distributed Learning, Intelligent Algorithm, Supervised Learning.

Abstract: Distributed systems require secure, flexible, and efficient access control mechanisms to protect their resources and data. Attribute-Based Access Control (ABAC) has been found to be suitable for dynamic and cooperative settings of distributed environments. The successful implementation of ABAC in any system requires the formulation of a complete and correct ABAC policy. Creating a policy for ABAC adoption requires a substantial amount of computation and administrative effort. The scale of computational requirements and administrative efforts is further magnified if the target system of deployment is distributed in nature. Several heuristic methods have been proposed for ABAC policy generation. The amount of resources and efforts that need to be invested in policy formulation can be substantially reduced by leveraging machine learning techniques. In this paper, we propose an intelligent framework for mining ABAC policies from access logs for distributed systems. The task of policy generation is carried out in two phases. In the first phase, an initial policy is created by each of the individual entities of the distributed system. In the second phase, all the individually created policies are combined together to create the final ABAC policy. The proposed framework ensures data privacy by preventing the need for an entity to share its access log with any other entity by leveraging Federated Learning (FL) to create the ABAC policy. Experimental results on three access control datasets show that our proposed strategy creates ABAC policies which can efficiently and effectively evaluate access requests and perform access decision inferencing.

1 INTRODUCTION

Distributed systems have become the backbone of modern computing, offering scalability, flexibility, and efficiency. Such systems often span multiple locations, data centers, cloud platforms, and interconnected services and allow remote access to resources. In a distributed environment, it is extremely essential that only authorized users are allowed access to resources and information in legitimate ways, while prohibiting all types of unauthorized accesses. Due to the dynamic and collaborative nature of distributed systems, ensuring secure access to resources poses significant challenges. Over the years, several access control models have been proposed, like Role-Based Access Control (RBAC) (Sandhu et al., 1996), Temporal RBAC (Bertino et al., 2001), Spatial RBAC (Damiani et al., 2007), and Rule-Based Access Control (Carminati et al., 2006). However, these access control models cannot effectively meet the flexible and diverse access requirements of distributed systems. Moreover, the dynamic nature of distributed systems makes the problem of access management and access administration a non-trivial one. In recent years, Attribute-Based Access Control (ABAC) model (Hu et al., 2013) has emerged as the de facto standard for access control provisioning. ABAC possesses the capability to address the unique challenges pertaining to access control and administration of distributed environments by bringing adaptability, context awareness, and granular control into access management, thereby making it quite suitable for distributed, collaborative and dynamic environments.

In ABAC, a user or subject is granted access to a resource or object based on the attributes of the user, the resource and several environmental conditions. An attribute of an entity is a specific property associated with that entity. Attributes are associated with subjects, objects and environmental context. Accordingly, they are referred to as subject attributes, object attributes and environmental attributes, respectively. Each attribute is either assigned a single value

Bai, S. K., Goveas, J. A. and Mitra, B.
Enhancing Access Control in Distributed Systems Through Intelligent ABAC Policy Mining.
DOI: 10.5220/0013636700003979
In *Proceedings of the 22nd International Conference on Security and Cryptography (SECRYPT 2025)*, pages 503-514
ISBN: 978-989-758-760-3; ISSN: 2184-7711

or multiple values from a predefined set of values. For ABAC model implementation, a set of rules is required. Each rule defines the required set of subject, object and optionally environmental attributes and the corresponding values in order to grant a subject access to an object. The set of all such rules constitutes an ABAC policy. To determine whether an access request is to be granted or denied, the attributes of the requesting subject, the requested object and optionally the environmental attributes and their values are taken into account. If each of the attributes has been assigned a specific value as per some rule, then the access request is granted. Otherwise, it is denied. Creating an ABAC policy ab initio is often done by considering the existing system access logs as inputs. Formulating an ABAC policy for a distributed environment requires taking into account the access logs of the individual entities/nodes of the environment. The final ABAC policy should reflect the access patterns generated for each of these entities/nodes. However, nodes may not be willing to share their access logs with any other node due to privacy concerns. Therefore, the primary challenge of policy creation in a distributed setup is to ensure the enumeration of a correct policy without sharing any information regarding the access patterns associated with the individual nodes.

In this paper, we propose an ABAC policy creation framework for distributed systems. Our framework generates a correct ABAC policy from the access logs of the different nodes of the distributed system without requiring any of the nodes to share their logs with any other node. We employ an intelligent technique to generate the ABAC policy using machine learning. The use of machine learning reduces the policy creation time as well as the time required to evaluate and make a decision for each access request. To ensure privacy of individual nodes, our framework uses Federated Learning (FL), a distributed machine learning paradigm, to create the final ABAC policy. Each node of the distributed system has access to this policy. To the best of our knowledge, this is the first effort to develop an intelligent and distributed ABAC policy learning framework.

The main contributions of the paper are summarized as follows:

- We propose a novel, intelligent ABAC policy generation framework for distributed systems. In this work, we have focused on a client-server like architecture.
- In our proposed framework, each client node acts as a *local policy creator*. A local policy creator generates an ABAC policy using a supervised machine learning algorithm by considering only its own access log. Such a policy is termed as a *local policy*. This is the first phase of our approach.
- The server node acts as the *global policy aggregator*. Once all the local policies are created, each local policy creator shares its local policy with the global policy aggregator. The global policy aggregator intelligently combines all the local policies to create the final, consolidated ABAC policy. This ABAC policy is shared with all the local policy creators. Each local policy creator evaluates the access requests based on this consolidated ABAC policy.
- Our proposed framework is privacy-preserving in nature since none of the local policy creators share their access logs with the global policy aggregator. Ensuring this privacy-preserving aspect of our framework as well as the integration of intelligence is done through the use of federated learning.
- Performance evaluation on three ABAC access request datasets show that our intelligent policy generation framework effectively creates an ABAC policy using which access decisions are made with a high degree of accuracy.

The rest of the paper is organized as follows. Section 2 reviews the existing literature on heuristic and intelligent techniques for ABAC policy mining and access control in distributed systems. Section 3 presents some preliminary concepts related to the ABAC model, distributed systems and federated learning. In Section 4, we describe our proposed distributed and intelligent policy creation framework. Section 5 presents the dataset description, experimental setup and the experimental results. We conclude the paper in Section 6 along with some insights about future research directions.

2 RELATED WORK

Policy mining aims to extract access control policies from existing data sources such as logs, access patterns, or configuration files. Automating policy creation significantly reduces the burden of system administrators, especially in large and complex systems. Several studies have investigated the generation and refinement of ABAC policies. Xu and Stoller in (Xu and Stoller, 2014) and (Xu and Stoller, 2015) present an algorithm for mining ABAC policies from logs and attribute data, highlighting the potential for automation in this domain. Talukdar et al. (Talukdar et al., 2017) develop ABAC-SRM, a bottom-up policy mining technique that can create generalized ABAC rules. Alternative methodologies, such as Rhapsody by Cotrini et al. (Cotrini et al., 2018), handle sparse inputs using rule reliability metrics. A top-down approach for automatic policy generation is proposed by the authors in (Anna Bamberger, 2024). This approach derives structured ABAC policies based on principal, resource, and environment attributes. In (Iyer and Masoumzadeh, 2018), the authors focus on the extraction of positive and negative authorization rules from access control information. Chaturvedi and Shirole (Chaturvedi and Shirole, 2024) discuss the privacy preservation of the attributes for an anonymous access. It introduces homomorphic attribute-based signature (HABS) module in the ABAC model. Nature inspired framework for policy mining is proposed in (Narouei and Takabi, 2019) where the authors present a methodology based on the particle swarm optimization algorithm.

Constrained policy mining can be used to specify additional conditions that must be met to grant or deny access. In (Gautam et al., 2017), a constrained policy mining algorithm is proposed, where the minimal set of ABAC rules is generated using the access control matrix as input and minimizing the weight of each rule and the sum of weights of all rules. Alohaly et al. (Alohaly et al., 2019) propose an automated framework for inferring ABAC constraints from natural language policies. The authors design an annotation scheme to capture ABAC constraint expressions within natural language access control policies (NLACPs) and a framework to extract the constraints from NLACPs.

Access control in distributed systems is a critical aspect of ensuring the security and integrity of data and resources in a network environment. Access control has many forms and mechanisms in a distributed environment, examples of which include Activity-Centric Access Control (ACAC) (Gupta and Sandhu, 2021), Extended Generalized Role-Based Access Control (EGRBAC) (Ameer et al., 2020), etc. Apart from these, Zhong et al. (Zhong et al., 2021) propose a distributed scheme for access control based on attribute-based encryption (ABE). In (Sikder et al., 2022), the authors propose a multi-user and multi-device-aware access control mechanism for shared smart home environments. Dynamic RBAC for decentralized applications (dApps) is a crucial aspect of a decentralized ecosystem. Chatterjee et al. (Chatterjee et al., 2020) present a framework for dynamic RBAC for decentralized applications where dApp is completely decoupled from the business applications.

The advancement of artificial intelligence and machine learning has greatly impacted the process of implementing access control with a focus on developing intelligent techniques for policy generation and administration. In ABAC systems, use of machine learning for policy learning, evolution, and enforcement enables adopting security mechanisms in dynamic environments as well as allowing adaptability. Researchers have explored various machine learning models for policy mining in access control systems. Restricted Boltzmann Machines (RBMs) have been employed for ABAC policy generation, as demonstrated in (Mocanu et al., 2015), where the authors propose an unsupervised learning approach to extract meaningful access control rules. Similarly, Karimi et al. (Karimi et al., 2022) introduce an automated ABAC policy learning framework leveraging unsupervised learning techniques to infer policies from access logs. The PAMMELA framework developed by Gumma et al. of (Gumma et al., 2022) employs supervised learning to deduce and augment ABAC rules. Polisma, proposed by Jabal et al. (Jabal et al., 2020), learns ABAC policies from access logs by using the combination of statistics, data mining and machine learning techniques. This work is extended by Jabal et al. in (Abu Jabal et al., 2023) by introducing FLAP, a collaborative policy learning approach designed for distributed environments. FLAP enables organizations to share policy knowledge and adapt policies based on insights derived from local logs, local policies, or hybrid learning techniques. In (Shan et al., 2024), the authors propose a deep learning-based method to generate ABAC policies from natural language documents, addressing the challenges of manual retrieval and analysis. The method uses the Chat General Language Model (ChatGLM) to extract access control-related statements from natural language documents, and the Iterated Dilated-Convolutions-Conditional Random Field (ID-CNN-CRF) model to annotate these attributes. Nobi et al. (Nobi et al., 2022b) propose Deep Learning Based Access Control (DLBAC). Reinforcement learning techniques, as demonstrated by (Cao et al., 2021) provide a novel way to address multi-user access control challenges in dynamic environments. Additionally, (Nobi et al., 2022a) explore machine learning-based access control methods that balance precision and scalability. These methods emphasize the increasing tendency to incorporate machine learning into access control systems to minimize administrative burden and improve flexibility.

Though a number of attribute-based access control mechanisms have been proposed for distributed systems, to the best of our knowledge, little effort has been made to develop an intelligent ABAC policy generation method for distributed systems. In this work, we address the challenge of mining ABAC policies in distributed environments without requiring any node of the distributed setup to disclose their individual access logs. We propose an intelligent, distributed and privacy-preserving ABAC policy learning framework by using federated learning.

3 PRELIMINARIES

In this section, we present the preliminary concepts related to the ABAC model, distributed systems and federated learning.

3.1 ABAC Model

The key components of the ABAC model (Hu et al., 2013) are the following:

- **Subjects (S):** A set of subjects S where a subject can be a human user or a system who/which requests an access.
- **Objects (O):** A set of objects O where an object refers to a system resource to which a subject requests an access.
- **Environmental Conditions (E):** The environment is the broader context of each access request. A set of environmental conditions E where each condition corresponds to a temporal, spatial or some other context in which a resource access occurs. Examples of such conditions are time, location, etc.
- **Attributes:** Attributes are characteristics associated with different entities in a system, such as users, resources, actions, and environments. These attributes can be both static and dynamic. The access policy of an organization is based on the attributes of the subject, object and environment involved in an access event. An attribute assumes value(s) from a set of attribute values. An attribute is said to be atomic/single valued if it is assigned a single value for a specific entity. A multi-valued attribute is assigned more than one value for a specific entity. Attributes can be of the following types.
  - **Subject Attributes (SA):** A set SA of subject attributes where each attribute defines a property of the corresponding subject. Examples of subject attributes include *age*, *job role*, *team membership*, *department*, *organization membership*, *management level*, etc.
  - **Object Attributes (OA):** A set OA of object attributes. Object attributes are the characteristics of an object. For eg. For a file object, attributes can be *name*, *size*, *creation date*, *owner* and *type*.
  - **Environmental Attributes (EA):** A set EA of environmental attributes. All environmental attributes depend on contextual factors such as the time and location of the access attempt. An environmental attribute can be *time of job shift* and the possible values can be *Morning Shift* and *Evening Shift*.
- **Permissions:** A set of permissions P that include operations like *read*, *write*, *update*, etc.
- **Policy:** A set of rules R that constitute the policy. Each rule of R defines if a particular access request involving certain subject, object and environmental attributes and the corresponding values is to be granted or denied. Rules can be of two types - positive rules and negative rules. A positive rule grants an access request and a negative rule denies an access request. An ABAC rule can be of the form $\langle\{$*Department = CSE, Role = Faculty, Course = DBMS*$\}$, $\{$*Type = Assignment, Department = CSE, Course = DBMS*$\}$, *evaluate*$\rangle$. This rule implies that a faculty of CSE department and associated with the DBMS course can evaluate the assignments of the same course offered by the CSE department.

3.2 Distributed Systems

Distributed system is a group of connected systems or computers that collaborate to deliver a service or solve a problem cooperatively. The systems or computers that are part of a distributed system are referred to as nodes. The goal of these systems is to increase performance by dividing processing responsibilities among several nodes, frequently spread across various physical locations and consequently optimizing resource utilization and performing load balancing. Moreover, distributed systems provide scalability, reliability and fault tolerance. Distributed Systems can be of two types (Liu and Antonopoulos, 2010), (Ricci and Carlini, 2012).

- **Client-Server Systems:** In such systems, multiple client nodes are connected to a server node. Each client sends a request for some service to the server and the server provides the corresponding service to the client. Usually, the clients communicate only with the server.
- **Peer-to-peer Systems:** These systems are decentralized in nature. None of the nodes are desig-

nated as a server or a client. Each node has equal responsibility and can function both as a service provider as well as a service requester.

In this work, we focus on client-server type distributed systems only. Distributed systems heavily rely on network based communications via message passing as well as data and resource sharing. These pivotal elements of distributed systems are necessary for their proper functioning, but they pose several security challenges by introducing various vulnerabilities into these systems. Managing secure and authorized access to data and resources in distributed environments is one such challenge which we have addressed in this work.

3.3 Federated Learning

Federated learning (McMahan et al., 2016) is a distributed approach to training machine learning models where the learning process occurs across multiple participants/nodes/clients without requiring any client to exchange or transmit any data. Each client (such as a mobile device or a computer) trains a model locally using only its own data. After the local model training is complete, each client shares only the model updates with a server. The server aggregates the model updates received from all the clients to create a global model. The global model is then transmitted to the clients. The clients again retrain the global model locally with their own data. This continues for a predefined number of rounds after which the final global model is obtained. Federated learning creates a model by considering the data from all the clients while preserving data privacy by eliminating the need for data transfer, and utilizing distributed data sources effectively. This is particularly useful in scenarios where data availability is fragmented and sensitive, while still allowing for collaborative learning and model improvement.

4 PROPOSED FRAMEWORK

We propose a distributed, learning-based policy mining framework which will enable system administrators to formulate ABAC policies in an automated manner. Our proposed framework will facilitate fast policy generation and rapid decision making for access requests in distributed systems. In this section, we present the system architecture for our framework and discuss the operation of our intelligent policy learning method.

4.1 System Architecture

We consider a distributed system consisting of n client nodes and a server node. Each client node communicates with the server node. However, the client nodes do not communicate among themselves. A client node is responsible for creating an ABAC policy based on its own individual access log. We refer to each such policy as a *local policy*. Thus, each client node acts as a *local policy creator*. The final, consolidated ABAC policy is created by the server by aggregating all the local policies. We refer to this final policy as the *global policy*. Therefore, the server acts as the *global policy aggregator*. In the rest of the paper, we refer to each client node as a local policy creator and the server node as the global policy aggregator.

The same set of attributes is associated with all the local policy creators. However, the set of values for each attribute may or may not be the same across the n local policy creators. Suppose, A is the set of all attributes (subject, object and environmental) across all the local policy creators and $A = \{attr_1, attr_2, \ldots, attr_k\}$. Let the set of values associated with some attribute $attr_i$ $(1 \leq i \leq k)$ be $V = \{v_1, v_2, \ldots, v_p\}$. Now, let us consider two local policy creators, LPC_x and LPC_y $(1 \leq x, y \leq n)$. $attr_i$ will be associated with both LPC_x and LPC_y. Let $V_{i,x}$ and $V_{i,y}$ denote the set of values for $attr_i$ associated with LPC_x and LPC_y respectively. $V_{i,x} \subseteq V$ and $V_{i,y} \subseteq V$. Also, either of the following is possible - (i) $V_{i,x} \cap V_{i,y} = \phi$, or (ii) $V_{i,x} \cap V_{i,y} \neq \phi$. This implies that sets of values for a particular attribute that is associated with different local policy creators may or may not have some common values. Our proposed framework is capable of handling both the scenarios.

Figure 1 shows the overall architecture of the proposed framework. In this figure, we have shown n local policy creators. Each local policy creator is equipped with the following.

- *Access Log:* Each local policy creator possesses its own access log using which it creates the local policy. In this work, we assume that the access log of each local policy creator is correct and does not contain any erroneous information (also termed as noise). Moreover, the access logs of the different local policy creators contain the same set of attributes. However, the sets of attribute values may be different.
- *Local Policy Repository:* Each local policy creator has a repository to store the locally generated policy as well as the final global policy. A local

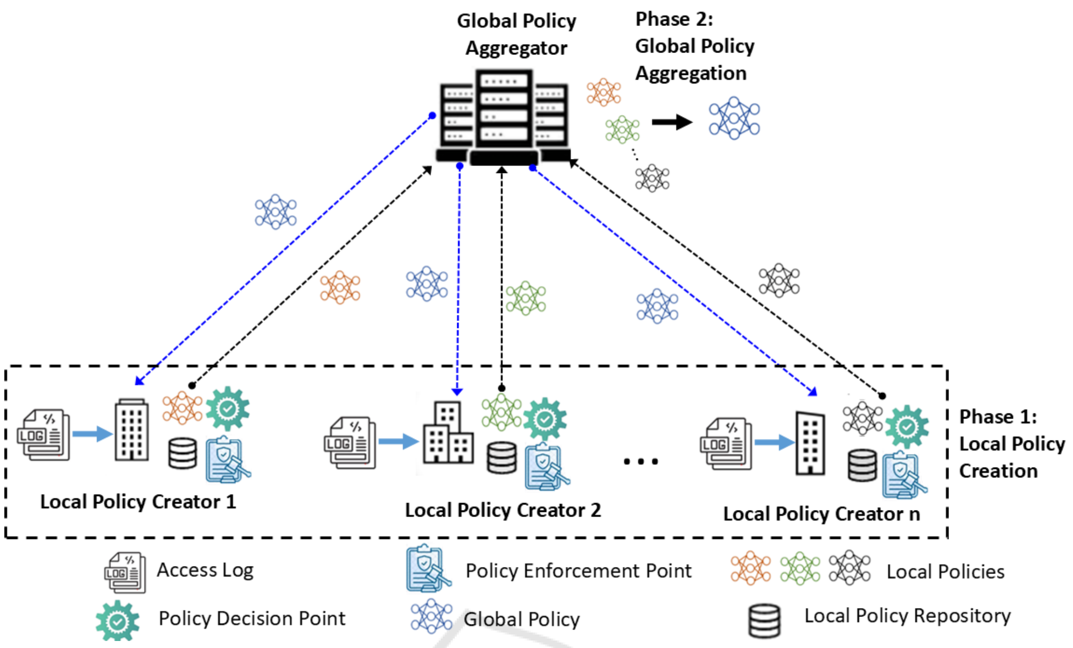

Figure 1: System Architecture and Operation of the Proposed Framework.

policy creator may choose to store both the local and the global policies in the repository or store only the global policy once it is finally created.

- *Policy Decision Point:* A policy decision point is associated with each local policy creator. When a subject requests to access an object, the request is forwarded to the policy decision point. The policy decision point evaluates the access request based on the final global policy. The subject is granted the access if it is permissible as per the global policy. Otherwise, the policy decision point denies the access request. The following points are to be noted in this context.
  - For each access request, the requesting subject and the requested object may be associated with the same local policy creator or may be associated with two distinct local policy creators. Our proposed framework is capable of handling both types of access requests.
  - Each object is associated with exactly one local policy creator. The same type of object may be present with two or more local policy creators. However, we consider these to be distinct and different instances of the same object type.
  - Access request for an object associated with a local policy creator is forwarded to and evaluated by the policy decision point of the local policy creator possessing the object.
- *Policy Enforcement Point:* After the policy decision point of a specific local policy creator has taken a decision regarding an access request, the policy enforcement point of that local policy creator makes sure that if the decision was to grant the access, then the requesting subject is allowed to access the requested object and if the decision was to deny the access, then the subject is prohibited from accessing the object.

4.2 Intelligent and Distributed ABAC Policy Mining

Our proposed framework for intelligent and distributed ABAC policy mining operates in two phases. We next discuss each of these phases in details.

Phase 1: The first phase is the *Local Policy Creation*. In this phase, each local policy creator generates an ABAC policy using a supervised machine learning algorithm. Each local policy is derived from the access log available with the corresponding local policy creator. We assume that such access logs are available with each local policy creator and the logs have been created as a result of past accesses. Each access log contains a set of access requests. An access request contains (i) the attributes and the corresponding values associated with the requesting subject, (ii) the attributes and the corresponding values associated with the requested object, (iii) the environmental attributes and their corresponding values, and (iv) the access decision, like grant or deny that was given for this access request in the past. Thus, each access log can be considered as a labeled dataset. All the local policy cre-

ators train a supervised machine learning model using their access logs. The same supervised learning algorithm is used by all the local policy creators. Phase 1 terminates when each of the local policy creators finish creating the local policies.
Phase 2: In phase 2, each local policy creator sends the locally generated policy to the global policy aggregator. The global policy aggregator combines the individual local policies to create the global policy. Once the global policy is created, the global policy aggregator shares it with all the local policy creators. This global policy is a trained machine learning model. Now, each local policy creator retrains this model on its own local access log. Once the retraining is complete, each local policy creator shares the updated policy (i.e., the updated machine learning model) with the global policy aggregator. This sequence of updating the policy locally by each local policy creator and the global policy aggregation continues for a pre-defined number of rounds after which the final, consolidated global policy is obtained. This final global policy is shared with all the local policy creators.

It is to be noted here that none of the local policy creators share their access logs with the global policy aggregator. Only the local policies are shared with the aggregator in the form of model updates. This becomes possible since our intelligent policy generation framework uses federated learning to derive the global policy. Consequently, our technique ensures the privacy of the individual local policy creators because the access patterns of the subjects are never shared with the aggregator. Moreover, since the local policies (initial or updated) are shared with the global policy aggregator as model updates, it is not possible for the aggregator to infer anything about the access patterns of the subjects from the local policies.

Once the global ABAC policy is generated, it is shared with the local policy creators. Additionally, the global policy is also stored with the global policy aggregator. This global policy is in the form of a trained machine learning model. Thus, the trained model acts as the policy decision point at each local policy creator. Consequently, the policy lookup time and the time required to evaluate access requests are substantially reduced when compared with heuristic methods, while automating the entire process. Moreover, the use of heuristic methods of distributed policy generation would have required either sharing the access logs with the aggregator or sharing the individual local policies in a human-readable format with the aggregator. Either of the scenarios would not have preserved the privacy of the access patterns of the subjects associated with the local policy creators. Our proposed framework eliminates the need of sharing the access logs or the local policies in human-readable format, thereby ensuring privacy of the subjects.

One point is to be noted here that the local policy creators do not share any model updates among themselves. Once the trained model is deployed as the policy decision point, the access requests are forwarded to it at each local policy creator. Such an access request can originate from the same local policy creator with which the requested object is associated or with a different local policy creator. Our proposed policy learning framework is capable of handling both types of access requests even if the trained model encounters some unseen attribute values in these requests. This is because the final ABAC policy is generated in a manner that it considers the access patterns of all the local policy creators and hence incorporates all values across all the attributes.

5 PERFORMANCE EVALUATION

In this section, we first present a description of the datasets that we have used to evaluate the performance of our proposed framework. We also describe our experimental setup and present the experimental results.

5.1 Dataset Description

For evaluating the performance of our intelligent and distributed policy formulation framework, we have used the three access control datasets presented in (Gumma et al., 2022). These datasets are named as *University Dataset 1*, *University Dataset 2* and *Company Dataset*. We have considered each of these datasets as access logs. The sizes of the University Dataset 1 and the Company Dataset were not sufficient to be used in a distributed setup. Hence, we have augmented these two datasets. We have also augmented the University Dataset 2. The augmentation is manifested in the form of more tuples/access requests being added to the datasets. The datasets have been augmented using the following strategies - (i) introducing new values for some attributes, and (ii) adding more access requests to the dataset by inferring all possible valid combinations of attribute values. The combinations that we have considered in (ii) have been done for both subject and object attributes. For eg., If we have two subject attributes, *Designation* having a possible value as *Accountant*, and *Department* having

Table 1: Details of Datasets used in Experiments.

Dataset	# Access Requests	Attribute-Name	Attribute-Type	# Attribute Values
University Dataset 1	8343 (#P = 1463, #N = 6880)	Designation	Subject	3
		Department	Subject	5
		Degree	Subject	3
		Year	Subject	4
		Resource-Type	Object	7
		Department	Object	5
		Degree	Object	3
		Year	Object	4
University Dataset 2	199835 (#P = 5858, #N = 193977)	Designation	Subject	5
		Post	Subject	6
		Course	Subject	120
		Department	Subject	5
		Degree	Subject	2
		Year	Subject	4
		Resource-Type	Object	8
		Department	Object	5
		Course	Object	120
		Degree	Object	2
		Year	Object	4
Company Dataset	41037 (#P = 3705, #N = 37332)	Designation	Subject	28
		Project-Name	Subject	10
		Department	Subject	8
		Resource-Type	Object	25
		Project-Name	Object	10
		Department	Object	7

two possible values as *Finance* and *Technical*, then $\langle Designation = Accountant, Department = Finance\rangle$ is a valid combination of subject attribute values, but $\langle Designation = Accountant, Department = Technical\rangle$ is not. It can be noted here that we have ensured that the datasets do not contain any duplicate tuples even after augmenttation. The details of the three datasets are presented in Table 1 in terms of the number of access requests (# Access Requests), the names of the different attributes (Attribute-Name), the type of each attribute (Attribute-Type), i.e, whether the attribute is a subject attribute or an object attribute and the number of values associated with each attribute (# Attribute Values). The second column (# Access Requests) also shows the number of positive access requests, i.e., the access requests for which the decision is grant (denoted by #P in the table column) and the number of negative access requests, i.e., the access requests for which the decision is deny (denoted by #N in the table column) for each dataset. It can be observed from the table that the number of negative access requests is much higher than the number of positive access requests in each dataset. This reflects a real-world scenario where out of the large number of access requests generated, only a handful are permissible and consequently, granted based on a pre-defined policy.

5.2 Experimental Setup

For our experiments, we have considered a setup consisting of four local policy creators and one global policy aggregator. The attributes are treated as features in our federated learning based setup. We use categorical data to create the feature vector to be given as input to the model. For each set of values associated with an attribute, the categorical encoding begins from 1 and keeps increasing monotonically thereafter. Each dataset is first split into the train set and test set with train set including 80% of the dataset tuples and the test set containing 20% of the tuples. The train set is used for policy mining and the test set is used to evaluate the performance of our framework. The train set is further partitioned into four segments. Each of these segments is assigned to a local policy creator node. We have experimented with three strategies for partitioning the train dataset. These are discussed are as follows.

- *Random Partitioning (RNP):* The train dataset is randomly partitioned into four parts and each part is assigned to one local policy creator. In this strategy, the dataset tuples (access requests) are randomly distributed across the clients. We do not attempt to explicitly keep track that how the values of the various attributes have been partitioned across the different local polciy creators.

- *Perception Based Partitioning (PBP):* We have carefully inspected each dataset and analytically inferred the attribute based on which the train data can be partitioned. The selection of this attribute is done across all the subject and object attributes. This strategy is purely based on human perception. For example, a system administrator can decide that the best way to partition the dataset is based on the subject attribute *Department*. Once the attribute is selected, the tuples present in the train set are grouped based on the values of this attribute. Thus, if $attr_i$ is selected and it has p values, then p partitions of the train data are created. After this, each of the p parts is equally divided among the n ($n = 4$ in our experiments) local policy creators.
- *Scoring Based Partitioning (SBP):* In this strategy, we use the SelectKBest feature selection technique available in the Scikit-learn library for Python. This technique evaluates each feature of the dataset using a scoring function and returns the top k features based on their scores. Once scoring is done, we select the best attribute/feature for each of the access control datasets. After this, a partitioning technique similar to the one followed in PBP is used to partition the train data across the local policy creators.

In a real-world setup, there will be no need to explicitly partition any dataset. Each local policy creator will be equipped with its own access request log. However, the nature of these local/individual datasets will vary from one distributed system to another in terms of the distribution of the attribute values. All the local policy creators will be aware of all the values corresponding to each attribute. However, each local policy creator's access log may or may not contain access request tuples for all the values of a particular attribute. Thus, the attribute values may be randomly or evenly distributed across the various local policy creators. To analyze the effect of the different distributions of the attribute values across the local nodes on the performance of our proposed framework, we have considered RNP, PBP and SBP. RNP corresponds to the random distribution of attribute values whereas PBP and SBP correspond to even distribution of attribute values. To study the effects of even distribution of attribute values, we have considered two types of partitioning strategies- one that is human perception driven and the other one that is model driven.

For University Dataset 1, 6674 access requests have been used for policy generation and 1669 access requests have been used for inferencing access decisions. In case of University Dataset 2, we have used 159868 tuples for policy creation and 39967 tuples have been used for inferencing. For the Company Dataset, policy creation has been carried using 32829 tuples and 8208 access requests are used for access decision inferencing. It is to be noted in this regard that policy creation constitutes the training phase of the federated learning model and access decision inferencing corresponds to the testing phase. Table 2 shows the details of partitioning each of the datasets across the four local policy creators (denoted as LPC_i such that i = 1, 2, 3, 4 in the table) for each partitioning strategy. The table depicts the number of training tuples used by each local policy creator for each dataset partitioning technique. The column labeled as PS represents the specific partitioning strategy. It can be observed from the table that the three partitioning methods tend to partition the train data almost similarly across the different local policy creators. The number of testing tuples is not affected by the partitioning method. For each local policy creator, the table further shows the number of positive access requests (#P) and the number of negative ac-

Table 2: Details of Training Data Partitioning for the three Partitioning Strategies.

PS	**University Dataset 1**			
	LPC_1	*LPC_2*	*LPC_3*	*LPC_4*
RNP	1668 (#P=319, #N=1349)	1668 (#P=285, #N=1383)	1668 (#P=304, #N=1364)	1670 (#P=277, (#N=1393)
PBP	1665 (#P=301, #N=1364)	1665 (#P=302, #N=1363)	1665 (#P=291, #N=1374)	1679 (#P=291, #N=1388)
SBP	1667 (#P=294, #N=1373)	1667 (#P=307, #N=1360)	1667 (#P=289, #N=1378)	1673 (#P=295, #N=1378)
PS	**University Dataset 2**			
	LPC_1	*LPC_2*	*LPC_3*	*LPC_4*
RNP	39967 (#P=1160, #N=38807)	39967 (#P=1189, #N=38778)	39967 (#P=1203, #N=38764)	39967 (#P=1147, (#N=38820)
PBP	39964 (#P=1168, #N=38796)	39964 (#P=1163, #N=38801)	39964 (#P=1156, #N=38808)	39976 (#P=1204, #N=38772)
SBP	39962 (#P=1193, #N=38769)	39962 (#P=1121, #N=38841)	39962 (#P=1199, #N=38763)	39982 (#P=1178, #N=38804)
PS	**Company Dataset**			
	LPC_1	*LPC_2*	*LPC_3*	*LPC_4*
RNP	8207 (#P=704, #N=7503)	8207 (#P=771, #N=7436)	8207 (#P=749, #N=7458)	8208 (#P=715, #N=7493)
PBP	8204 (#P=739, #N=7465)	8204 (#P=709, #N=7495)	8204 (#P=734, #N=7470)	8217 (#P=757, #N=7460)
SBP	8195 (#P=758, #N=7437)	8195 (#P=730, #N=7465)	8195 (#P=702, #N=7493)	8244 (#P=749, #N=7495)

cess requests (#N) in both the train and test sets. For University Dataset 1, the testing data contains 278 positive access requests and 1391 negative access requests. For University Dataset 2, the test data contains 1159 positive access requests and 38808 negative access requests. The test data of Company Dataset contains 766 positive access requests and 7442 negative access requests.

We have implemented our proposed approach using Pytorch 2.5.1. We have used Artificial Neural Network (ANN) as the supervised machine learning model running at each local policy creator. We have trained our model using a batch size of 32, hidden layer size of 30, a learning rate of 0.001, Adam optimizer and Binary Cross-Entropy Loss. For local policy creation, the model runs for 10 epochs and the overall global policy creation takes place across 50 federation rounds. The global policy creator uses FedAvg to aggregate the local policies. FedAvg creates the global policy by averaging the individual model weights corresponding to the local policies. The experimentation was done on a desktop computer having 2.10 GHz, 20 cores, Intel core i7 processor, 16 GB RAM and Ubuntu 22.04 as the operating system.

5.3 Results

We evaluate the performance of our proposed framework with respect to its ability to provide correct decision for the incoming access requests in terms of the following metrics - Accuracy, Precision, Recall, aF1-score. We discuss these metrics as follows.

- *True Positive Access Requests (TPAR)*: These are the positive access requests (ones which should be permitted) for which our framework granted the access, thereby indicating accurate classification.
- *True Negative Access Requests (TNAR)*: These are the negative access requests (ones which should be denied) for which our framework denied the access, also representing accurate classification.
- *False Positive Access Requests (FPAR)*: These are the negative access requests for which our method incorrectly granted the access, leading to a security breach.
- *False Negative Access Requests (FNAR)*: These are the positive access requests which were denied by our method, making the system overly restrictive.

Accuracy denotes the percentage of the access requests (both positive and negative) for which correct decisions are taken. Precision measures the percentage of the granted access requests that are actually correct. Recall is the percentage of the positive access requests that are correctly classified. F1-score is the harmonic mean of precision and recall. These four metrics are calculated as follows:

$Accuracy = \frac{TPAR+TNAR}{TPAR+FPAR+TNAR+FNAR}$,

$Precision = \frac{TPAR}{TPAR+FPAR}$,

$Recall = \frac{TPAR}{TPAR+FNAR}$,

$F1-score = \frac{2*Precision*Recall}{Precision+Recall}$.

Table 3 shows the performance of our proposed framework in terms of the four evaluation metrics across the three train data partitioning strategies for the three datasets. In this table, *Acc*, *Pr*, *Rec* and $F1-s$ correspond to accuracy, precision, recall and F1-score respectively. For University Dataset 1, SBP gives the best performance for all the four metrics. In case of University Dataset 2, PBP and SBP perform more or less similarly, with both outperforming RNP. For the Company Dataset, RNP gives the best results. However, the performance of the three partitioning methods is within less than 2% margin of each other. The results show that our proposed framework is capable of accurately classifying the access requests by achieving an accuracy of more than 90% for all the datasets. It can also be seen that the precision is close to (or sometimes higher than) 95% for the University Datasets and is more than 85% for the Company Dataset. This implies that our policy learning framework can effectively prevent security breaches. Our proposed strategy gives somewhat lower performance for the Company Dataset in comparison to the two University Datasets mostly because of the large number of values present for the different attributes. The small percentage of the access requests for which our framework cannot prevent the security breaches can be manually inspected and decided by the system administrators associated with the local policy creator nodes. The recall values for the three datasets are lower than the corresponding precision values, thereby implying that, in certain cases, our approach may be somewhat over-restrictive. Overall, it can be concluded that our method can correctly evaluate access requests in an automated manner while requiring a very small degree of manual intervention to prevent certain security breaches.

Table 4 shows the total model training time over 50 federated rounds and the average access request inferencing time for the test data for each of the partitioning strategies for the three datasets in milliseconds. The training time corresponds to the time required to create the final ABAC policy and the average access inference time denotes the time required,

Table 3: Performance Evaluation of Our Proposed Framework.

PS	**University Dataset 1**				**University Dataset 2**				**Company Dataset**			
	Acc	*Pr*	*Rec*	*F1-s*	*Acc*	*Pr*	*Rec*	*F1-s*	*Acc*	*Pr*	*Rec*	*F1-s*
RNP	97.51	95.59	92.84	94.16	99.62	94.07	93.30	93.38	91.26	85.45	83.65	84.38
PBP	97.11	94.68	91.81	93.19	99.82	96.22	97.92	97.03	90.25	84.03	81.22	82.47
SBP	97.66	96.00	93.12	94.50	99.80	95.60	98.11	96.76	90.08	83.75	81.10	82.16

Table 4: Total Training and Average Access Inference Time (in milliseconds) of Our Proposed Framework.

PS	**University Dataset 1**		**University Dataset 2**		**Company Dataset**	
	Train Time	*Inference Time*	*Train Time*	*Inference Time*	*Inference Time*	*Inference Time*
RNP	12590	0.026	343930	0.028	59879	0.025
PBP	12570	0.025	343000	0.024	61079	0.026
SBP	12580	0.026	342580	0.026	59079	0.028

on an average, to evaluate and provide decision for a single access request. The results show that our proposed framework derives the ABAC policy very less time. This implies that in a real-world scenario, our framework will formulate the policy in a reasonably less amount of time thereby ensuring less amount of system downtime. However, in a real-world setup, the policy creation time will be higher due to the latency of the communications occurring between the local policy creator and the global policy aggregator.

The overall policy creation time is governed by the size of the dataset. For this reason, the policy generation time for University Dataset 2 is the highest followed by the Company Dataset and the University Dataset 1 having the lowest policy formulation time. Additionally, the time required to grant or deny an access request is quite low. Thus, our model will provide extremely fast decisions for incoming access requests even in a distributed setup. It can be seen from the table that the manner of distribution of the access tuples across the different local policy creators does not affect the model execution time. Thus, the overall performance results show that our proposed intelligent policy formulation framework is capable of accurately making access decisions in a short span of time in a distributed environment.

6 CONCLUSIONS

In this paper, we have proposed a distributed ABAC policy generation framework that leverages federated learning to intelligently create policies. Our framework operates in two phases. In the first phase, individual policies are created by the local policy creators using their own access logs. In the second phase, the local policies are aggregated by the global policy aggregator after which the global policy is shared with the local policy creators who in turn finetune the policy further. The second phase executes for a predefined number of federated rounds. Experimental results show that our proposed framework provides a high degree of performance in terms of providing decision regarding access requests in a very short time duration. Moreover, the policy creation time is quite less and can be suitably deployed in a real-world scenario.

In future, we intend to use tree-based classifiers to perform policy mining for distributed systems in a privacy-preserving manner. We also plan to use deep learning and reinforcement learning to create ABAC policies for distributed environments. The objective will be to design and develop an accurate and lightweight policy decision engine that can be deployed even in resource-constrained distributed systems. It will also be interesting to analyze to what extent intelligent policy mining techniques are resilient to the presence of noise in the input access logs. Another future direction of research that we plan to undertake is to analyze the extent to which communication latencies in a distributed setup affects the time required to make access control decisions.

REFERENCES

Abu Jabal, A., Bertino, E., Lobo, J., Verma, D., Calo, S., and Russo, A. (2023). Flap - a federated learning framework for attribute-based access control policies. In *13th ACM Conference on Data and Application Security and Privacy*, page 263–272.

Alohaly, M., Takabi, H., and Blanco, E. (2019). Towards an automated extraction of abac constraints from natural language policies. In *ICT Systems Security and Privacy Protection*, pages 105–119.

Ameer, S., Benson, J., and Sandhu, R. (2020). The egrbac model for smart home iot. In *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 457–462.

Anna Bamberger, M. F. (2024). Automated generation and update of structured abac policies. In *ACM Workshop on Secure and Trustworthy Cyber-Physical Systems*, page 31–40.

Bertino, E., Bonatti, P. A., and Ferrari, E. (2001). Trbac: A temporal role-based access control model. *ACM Transaction on Information & System Securiry*, 4(3):191–233.

Cao, Y., Lien, S.-Y., and Liang, Y.-C. (2021). Deep reinforcement learning for multi-user access control in non-terrestrial networks. *IEEE Transactions on Communications*, 69(3):1605–1619.

Carminati, B., Ferrari, E., and Perego, A. (2006). Rule-based access control for social networks. In *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, pages 1734–1744.

Chatterjee, A., Pitroda, Y., and Parmar, M. (2020). Dynamic role-based access control for decentralized applications. In *ICBC 2020*, page 185–197.

Chaturvedi, G. K. and Shirole, M. (2024). Abac-pa2: Attribute-based access control model with privacy aware anonymous access. *Procedia Computer Science*, 237:147–154.

Cotrini, C., Weghorn, T., and Basin, D. (2018). Mining abac rules from sparse logs. In *2018 IEEE European Symposium on Security and Privacy*, pages 31–46.

Damiani, M. L., Bertino, E., Catania, B., and Perlasca, P. (2007). Geo-rbac: A spatially aware rbac. *ACM Transactions on Information and System Security*, 10:2–es.

Gautam, M., Jha, S., Sural, S., Vaidya, J., and Atluri, V. (2017). Constrained policy mining in attribute based access control. In *22nd ACM on Symposium on Access Control Models and Technologies*, page 121–123.

Gumma, V., Mitra, B., Dey, S., Patel, P. S., Suman, S., Das, S., and Vaidya, J. (2022). Pammela: Policy administration methodology using machine learning. In *19th International Conference on Security and Cryptography - SECRYPT*, pages 147–157.

Gupta, M. and Sandhu, R. (2021). Towards activity-centric access control for smart collaborative ecosystems. In *26th ACM Symposium on Access Control Models and Technologies*, page 155–164.

Hu, V. C., Ferraiolo, D., Kuhn, R., Friedman, A. R., Lang, A. J., Cogdell, M. M., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K., et al. (2013). Guide to attribute based access control (abac) definition and considerations (draft). *NIST special publication*, 800:1–54.

Iyer, P. and Masoumzadeh, A. (2018). Mining positive and negative attribute-based access control policy rules. In *23nd ACM on Symposium on Access Control Models and Technologies*, page 161–172.

Jabal, A. A., Bertino, E., Lobo, J., Law, M., Russo, A., Calo, S., and Verma, D. (2020). Polisma - a framework for learning attribute-based access control policies. In *European Symposium on Research in Computer Security*, volume 12308, pages 523–544.

Karimi, L., Aldairi, M., Joshi, J., and Abdelhakim, M. (2022). An automatic attribute-based access control policy extraction from access logs. *IEEE Transactions on Dependable and Secure Computing*, 19:3–16.

Liu, L. and Antonopoulos, N. (2010). *From Client-Server to P2P Networking*, pages 71–89. Springer US, Boston, MA.

McMahan, B., Moore, E., Ramage, D., Zhang, S., Léon, D., and Snoek, M. (2016). Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*.

Mocanu, D., Turkmen, F., and Liotta, A. (2015). Towards abac policy mining from logs with deep learning. In *18th International Multiconference, IS 2015, Intelligent Systems*.

Narouei, M. and Takabi, H. (2019). A nature-inspired framework for optimal mining of attribute-based access control policies. In *Security and Privacy in Communication Networks*, pages 489–506. Springer International Publishing.

Nobi, M. N., Krishnan, R., Huang, Y., and Sandhu, R. (2022a). Administration of machine learning based access control. In *Computer Security – ESORICS 2022: 27th European Symposium on Research in Computer Security,Part II*, pages 189–210.

Nobi, M. N., Krishnan, R., Huang, Y., Shakarami, M., and Sandhu, R. (2022b). Toward deep learning based access control. In *12th ACM Conference on Data and Application Security and Privacy*, page 143–154.

Ricci, L. and Carlini, E. (2012). Distributed virtual environments: From client server to cloud and p2p architectures. In *2012 International Conference on High Performance Computing & Simulation (HPCS)*, pages 8–17.

Sandhu, R., Coyne, E., Feinstein, H., and Youman, C. (1996). Role-based access control models. *Computer*, 29:38–47.

Shan, F., Wang, Z., Liu, M., and Zhang, M. (2024). Automatic generation of attribute-based access control policies from natural language documents. *Computers, Materials and Continua*, 80:3881–3902.

Sikder, A. K., Babun, L., Celik, Z. B., Aksu, H., McDaniel, P., Kirda, E., and Uluagac, A. S. (2022). Who's controlling my device? multi-user multi-device-aware access control system for shared smart home environment. *ACM Trans. Internet Things*, 3.

Talukdar, T., Batra, G., Vaidya, J., Atluri, V., and Sural, S. (2017). Efficient bottom-up mining of attribute based access control policies. In *2017 IEEE 3rd International Conference on Collaboration and Internet Computing*, pages 339–348.

Xu, Z. and Stoller, S. D. (2014). Mining attribute-based access control policies from logs. In *Data and Applications Security and Privacy XXVIII*.

Xu, Z. and Stoller, S. D. (2015). Mining attribute-based access control policies. *IEEE Trans. Dependable Secur. Comput.*, 12:533–545.

Zhong, H., Zhou, Y., Zhang, Q., Xu, Y., and Cui, J. (2021). An efficient and outsourcing-supported attribute-based access control scheme for edge-enabled smart healthcare. *Future Generation Computer Systems*, 115:486–496.